

JS Level 0



Практикуемся



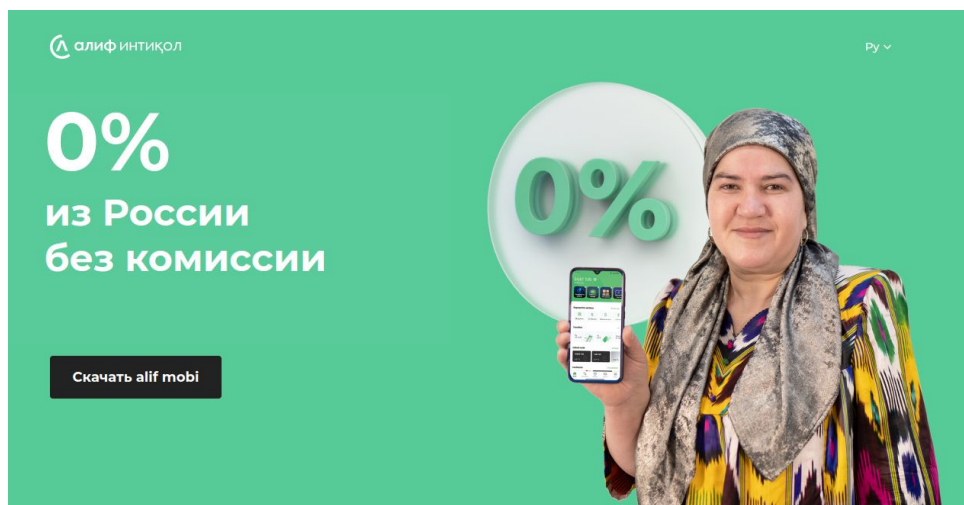
Практикуемся

В этой лекции мы применим на практике уже имеющиеся у нас знания (и получим новые) на примере существующего веб-сайта.

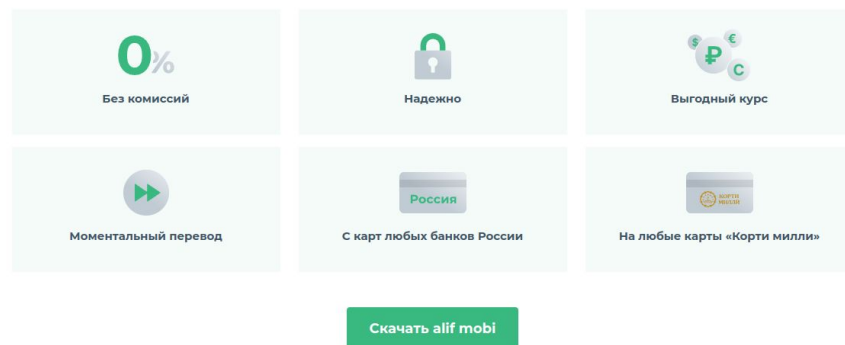


Практикуемся

Возьмём за основу следующую страничку (<https://intiqol.tj>):



Преимущества alif mobi



Практикуемся

Как всегда, есть куча подходов, как это делать. Какие CSS-селекторы использовать, как их называть и т.д.

Мы выделим два подхода:

- оформительский, когда вы элементам присваиваете классы исходя из их внешнего вида, например: `class="big green"` (означает, что это какой-то большой и зелёный элемент, обратите внимание: классы разделены, т.е. может быть где-то большой, но жёлтый элемент)
- смысловой, когда вы элементам присваиваете классы исходя из их предназначения (например, `class="register_btn"`, кнопка регистрации).



Практикуемся

Попробуем применить к нему стилизацию, используя второй подход (смысловый).

Первое, с чего мы начинаем, это выделяем крупные блоки нашей страницы и присваиваем им классы (см. следующий слайд).



Практикуемся

```

<body>
  <header class="header">
    <nav class="nav">
      <a href="https://alif.tj" class="logo">
        
      </a>
    </nav>
    <div class="promo">
      <p class="keypoint">
        <span class="comission">0%</span>
        <span class="description">из России<br>без комиссии</span>
      </p>
      <a href="https://app.adjust.com/bak18ku?adgroup=Bunner" class="button-primary">Скачать alif mobi</a>
    </div>
  </header>
  <main class="main">
    <section class="advantages"> ...
  </section>
  <section class="instruction"> ...
  </section>
  <section class="transfer"> ...
  </section>
</main>
  <footer class="footer"> ...
</footer>
</body>

```



Практикуемся

Почему классы? Общепринято, чтобы не разводить "зоопарк" для оформления использовать только классы (ни `id`, ни теги), а именно классы. Кроме того, использование классов позволит нам безболезненно заменять теги (и даже переехать с семантической вёрстки на `div` и обратно).

Поэтому мы распределяем классы.

Начнём с самых верхнеуровневых – `header` и `footer`.



Практикуемся

```
*, *::after, *::before {
  box-sizing: border-box;
}

body {
  font-family: 'Roboto', sans-serif;
  font-size: 16px;
}

.header {
  background-color: #56cb97;
  color: #ffffff;
}

.footer {
  background-color: #f7f8f9;
  color: #222222;
}
```

Вы можете сказать, но зачем `header`'у давать класс `header`, а `footer`'у класс `footer`?

Дело в том, что `header` может использоваться и внутри других элементов (а не только внутри `body`). Выдавая ему простой класс `header`, мы сами с собой договариваемся, что этот класс будет обозначать блок на верхнем уровне страницы.

И сразу мы можем эти классы стилизовать, выдав им цвета.



Забегая вперёд

Вы столкнётесь с этим только после первого своего сайта, но мы заранее вам расскажем общую историю. Если мы внимательно посмотрим на сайт, то увидим, что на нём есть несколько ключевых цветов:

- зелёный (фоновый – `#56CB97` и `#39B980` – цвет фона кнопки, текста ссылок и ключевых элементов – порядкового номера в списке и т.д.)
- чёрный (`#222222`* цвет обычного текста и фона выделенной кнопки)

Мы, конечно, можем их прописывать "как есть", Но что если завтра (через неделю, месяц) цвет поменяется? Мы, конечно, можем пройтись по всему файлу и заменить через поиск и замену всё. Но это оч.неудобно. Поэтому мы с вами сразу будем использовать современный подход.

Примечание*: кстати, если цифры в цвете повторяются `#RRGGBB`, то можно записывать только по одной цифре: `#RGB` (т.е. `#222222` -> `#222`), но лучше использовать все 6, т.к. тогда проще будет искать по файлу.



Забегаая вперёд

Современный подход заключается в том, что дизайнеры сразу определяют тему: это набор основных цветов, размеров элементов и т.д.

Не все банки это публикуют в открытом доступе, но вот, например, из документации [Альфа Банка](#):

Основные цвета

Pantone

RGB

CMYK

HEX

Alfa Red

#EF3124

Full Black

#000000

Graphite Gray

#505759

Cool Grey

#D9D9D6



Переменные в CSS

В CSS есть возможность задавать переменные - это просто имена, которые мы придумываем для каких-то значений, а потом использовать эти имена везде.

Вот как это выглядит:


```
:root {  
  --regular-text-color: ■ #222222;  
  --regular-fg-color: ■ #222222;  
  --regular-bg-color: □ #f7f8f9;  
  
  --primary-fg-color: □ #ffffff;  
  --primary-text-color: ■ #222222;  
  --primary-bg-color: ■ #222222;  
  
  --accent-fg-color: □ #ffffff;  
  --accent-text-color: ■ #39b980;  
  --accent-bg-color: ■ #56cb97;  
}
```



Переменные в CSS

Давайте разбирать по частям:

специальный псевдо-элемент, в который складывают переменные
значение переменной



```
:root {  
  --regular-text-color: ■ #222222;
```

имя переменной, должно начинаться с --

Т.е. мы придумали, что далее везде в наших стилях будем цвет #222222 называть именем `--regular-text-color`.



Переменные в CSS

Чтобы использовать это имя (т.е. значение, хранящееся в нём), мы вместо `#222222` пишем `var(--regular-text-color)`:

```
body {  
  font-family: 'Roboto', sans-serif;  
  font-size: 16px;  
  color: var(--regular-text-color);  
}
```

Это позволяет нам в одном месте написать имя и его значение и использовать во многих местах (поскольку слова и их сочетания, мы запоминаем лучше, чем числа вроде `#222222`).



Переменные в CSS

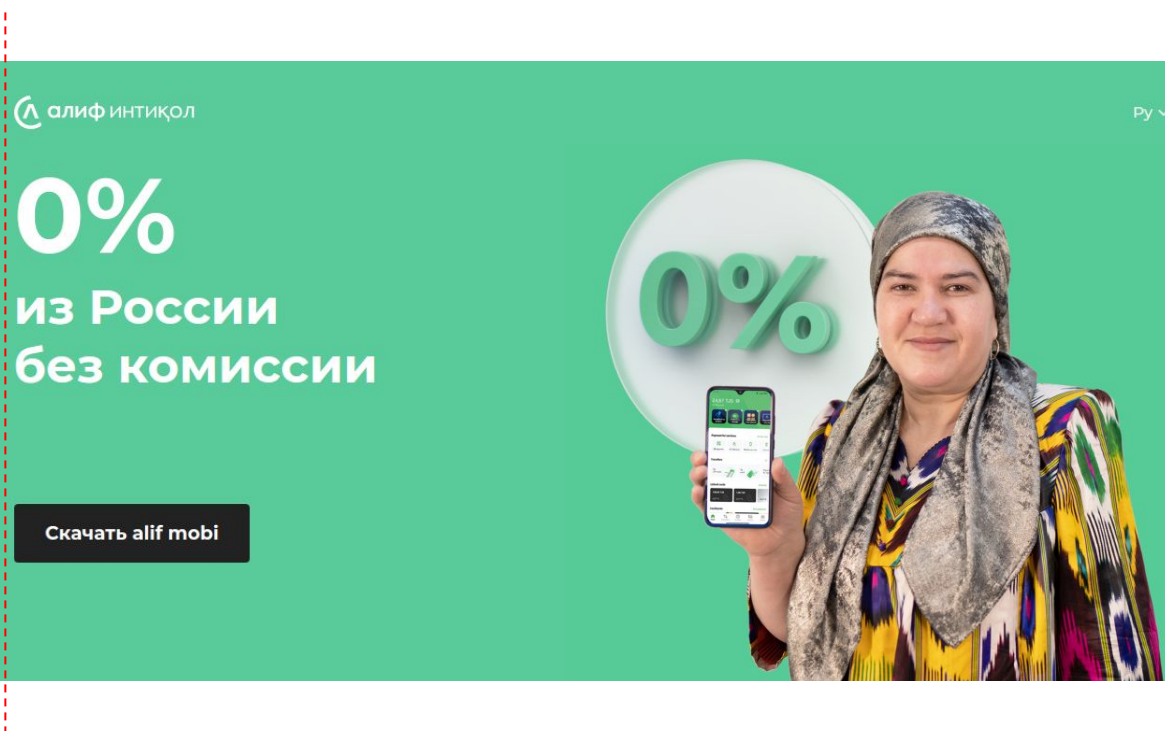
Мы можем задавать не только цвета в переменных, но чаще всего именно цвета задаются.

Здесь стоит отметить, что не всегда дизайнеры следуют этим соглашениям и выделяют тему. В этом случае вы можете обсудить этот вопрос с ними (это сделает вашу же работу комфортнее).



Container

Дальше нужно разобраться с тем, что весь контент на странице отцентрирован и не занимает всей ширины экрана (а фон - занимает):



Container

Для реализации подобного поведения часто вводят дополнительный `div`, который и выравнивает содержимое:

```
<header class="header">
  <div class="container">
    <nav class="nav">
      <a href="https://alif.tj" class="logo">
        
      </a>
    </nav>
    <div class="promo">
      <p class="keypoint">
        <span class="comission">0%</span>
        <span class="description">из России<br>без комиссии</span>
      </p>
      <a href="https://app.adjust.com/bak18ku?adgroup=Bunner" class="button-primary">Скачать alif mobi</a>
    </div>
  </div>
</header>
```



Container

В таблицах стилей наиболее общие стили (те, которые используются в разных частях документа) выносятся наверх:

```
.container {  
  padding: 0 15px;  
  margin: 0 auto;  
  max-width: 1024px;  
}  
  
.header {  
  background-color: var(--accent-bg-color);  
  color: var(--accent-fg-color);  
}
```



padding

`padding: 0 15px;` – это сокращённая запись:

`padding-top: 0;`

`padding-right: 15px;`

`padding-bottom: 0;`

`padding-left: 15px;`

Поскольку значения у верхнего и нижнего `padding`'а – `0`, то единицу измерения можно не писать (т.к. `0` – он в любой единице измерения будет `0`).



margin

`margin: 0 auto;` – это сокращённая запись:

`margin-top: 0;`

`margin-right: auto;`

`margin-bottom: 0;`

`margin-left: auto;`

Значение `auto` позволяет выставить автоматически рассчитанные отступы слева и справа (исходя из ширины элемента, максимальную мы задали с помощью `max-width: 1024px`).



background-image

Выставить размеры текста и отступы вы сможете сами, самое же главное для нас – как отобразить картинку. Если мы подумаем – то это изображение фоновое, значит, для его отображения не нужен элемент `img`.

Значит, мы должны выставить его с помощью CSS, а конкретно с помощью свойства `background-image` (или `background`, которое позволяет задать сразу несколько значений).



background-image

```
.header .promo {  
  background: url('https://alif-skills.pro/media/intiqol-promo.png') right bottom no-repeat;  
}
```

<https://drafts.csswg.org/css-backgrounds-3/#propdef-background>

По частям:

- **url** – функция, которая подставляет изображение по URL'у
- **right bottom** – правый нижний угол (см. <https://drafts.csswg.org/css-backgrounds-3/#typedef-bg-position>, можно указывать в %)
- **no-repeat** – не повторять (по умолчанию, повторяется, "закрывая" всю область)



background-image

Но если мы посмотрим на то, что получится, то получится не очень:



0% из России
без комиссии

[Скачать alif mobi](#)

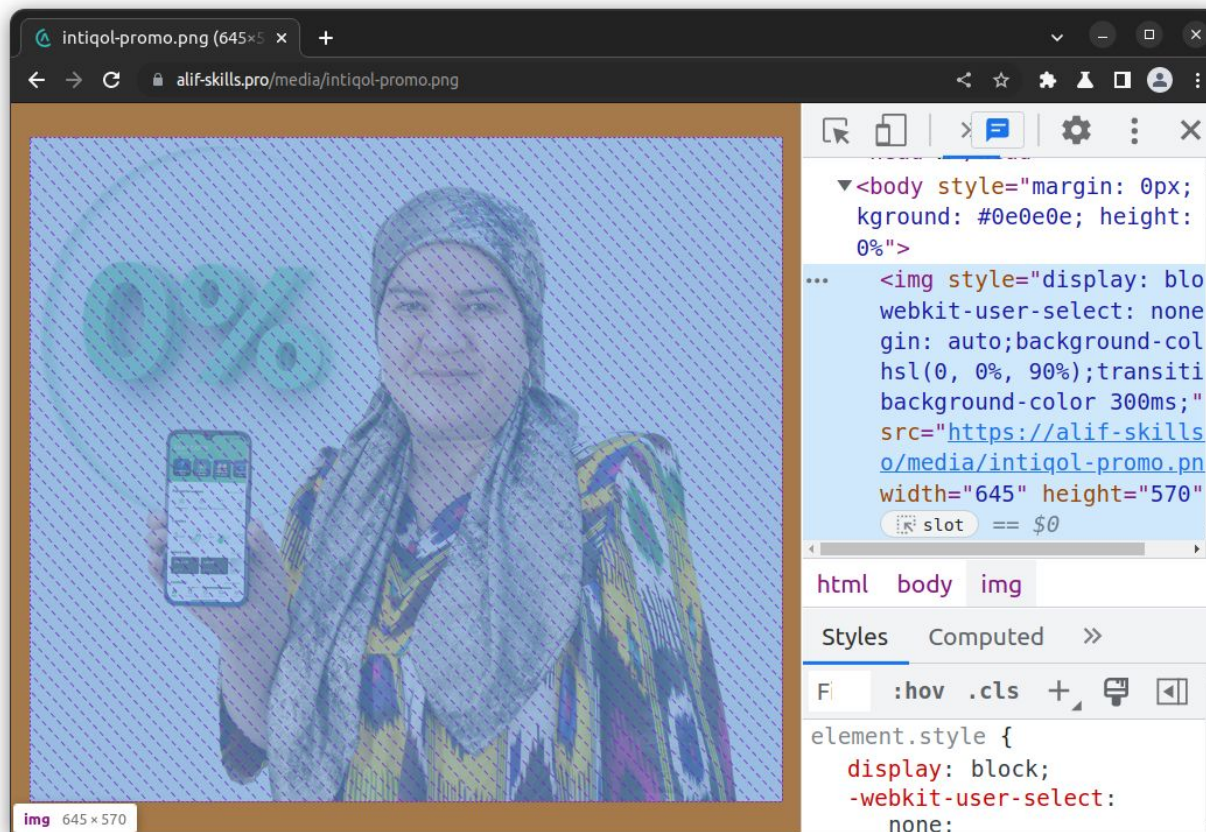


Мы специально не выставляли отступы, чтобы показать, что картинка будет обрезаться.



background-image

Чтобы она не обрезалась, нужно `.header.promo` выставить такую минимальную высоту (обратите внимание: минимальную, а не фиксированную), чтобы картинка точно умещалась. Для этого нужно знать размеры картинки:



background-image

```
.header .promo {  
  min-height: 570px;  
  background: url('https://alif-skills.pro/media/intiqol-promo.png') right bottom no-repeat;  
}
```



Кнопки

Поскольку кнопки встречаются повсеместно в нашем макете, имеет смысл их вынести за пределы объявлений конкретных блоков наших документов (например, `.promo` мы указывали как `.header .promo` – показывая с помощью составного селектора мы стилизуем именно `.promo` внутри `.header`).

Кнопки мы сгруппируем в две категории:

- `primary` – ключевую кнопку (чёрная в макете)
- `regular` – обычную кнопку (зелёная в макете)



Кнопки

```
.primary-btn {  
  display: inline-block;  
  padding: 15px 30px;  
  border: 0;  
  border-radius: 4px;  
  font-size: 18px;  
  font-family: inherit;  
  text-decoration: none;  
  color: var(--primary-fg-color);  
  background: var(--primary-bg-color);  
}
```

Ключевой вопрос: зачем мы выставляет `font-family`? Дело в том, что у кнопок (не `a`, а `button`) есть user agent style, что не позволяет наследовать шрифт от `body`.

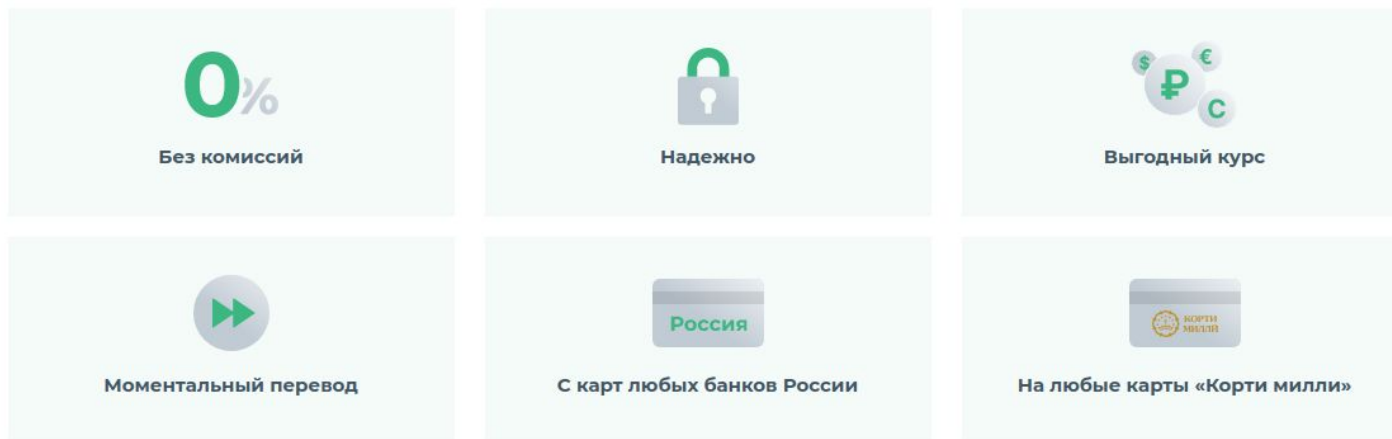
Это то, почему важно понимать вопросы наследования.



Преимущества

С преимуществами уже сложнее:

Преимущества alif mobi



Мы, конечно, можем подбирать размеры и т.д., но существуют более интересные техники.



Flexbox

Долгое время для построения интерфейса использовалась комбинация из `display`, `position` и `float` (это обтекание, мы не проходим, поскольку есть более удобные альтернативы).

Но это было сродни "колдовству" – установка каких-то непонятных комбинаций свойств для получения нужного результата.

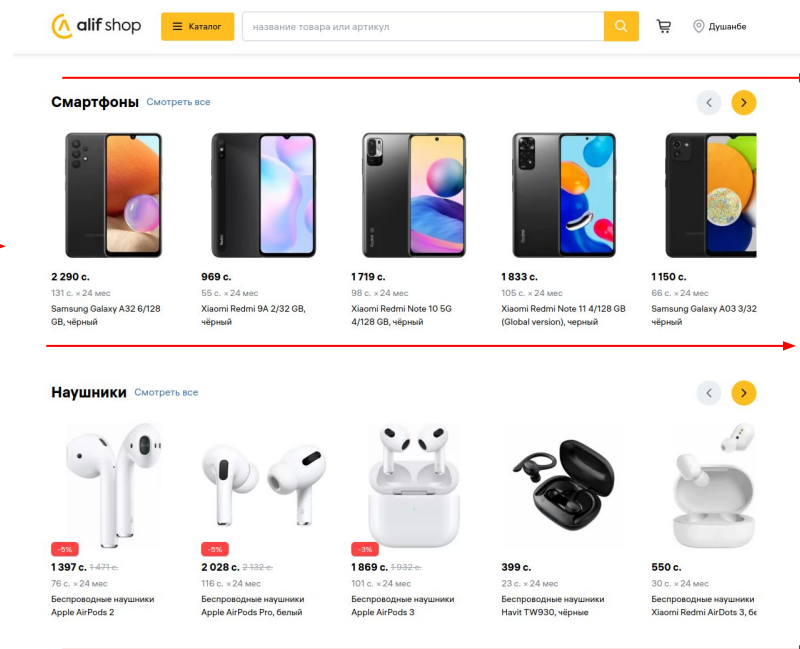
Поэтому, ожидаемо, появились другие системы, которые позволяют работать проще и эффективнее (но `display` и `position` при этом никуда не деваются).



Flex

Если мы внимательно посмотрим на многие сайты, то выяснится, что все элементы можно разложить на секции, упорядоченные по одной из осей (чаще всего это горизонтальная ось):

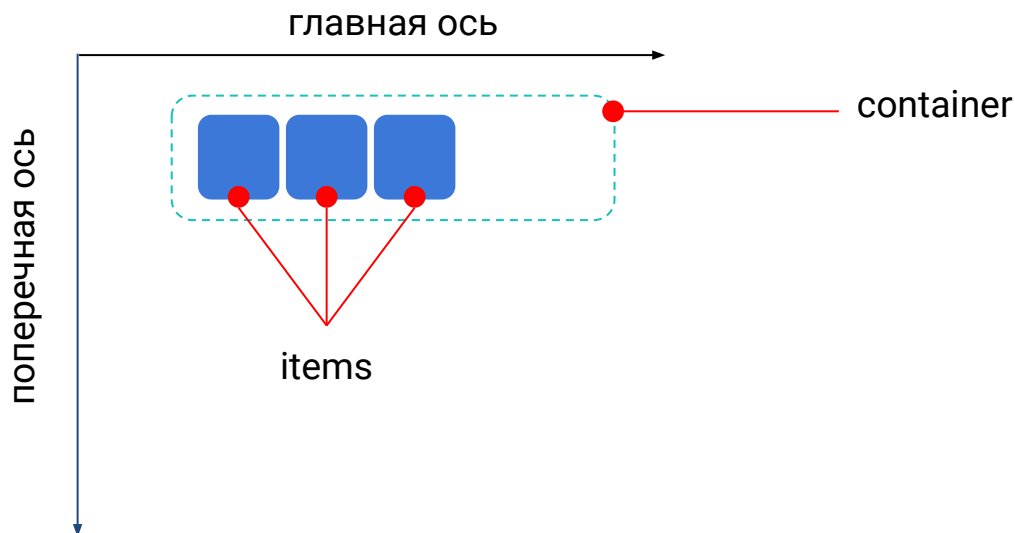
Именно для этих целей придумали специальную технологию, которая называется flexbox, которая позволяет гибко управлять размещением элементов в двух осях.



container и item

В рамках этой технологии используется три ключевых термина:

1. Container (контейнер) – элемент, внутри которого мы хотим гибко управлять размещением дочерних элементов
2. Items (элементы) – элементы, которые находятся внутри контейнера и размещением которых мы хотим управлять
3. Оси: главная и поперечная



Flex

В чём идея? Flex позволяет управлять дочерними элементами (их размерами и расположением), выстраивая их в этих двух осях.

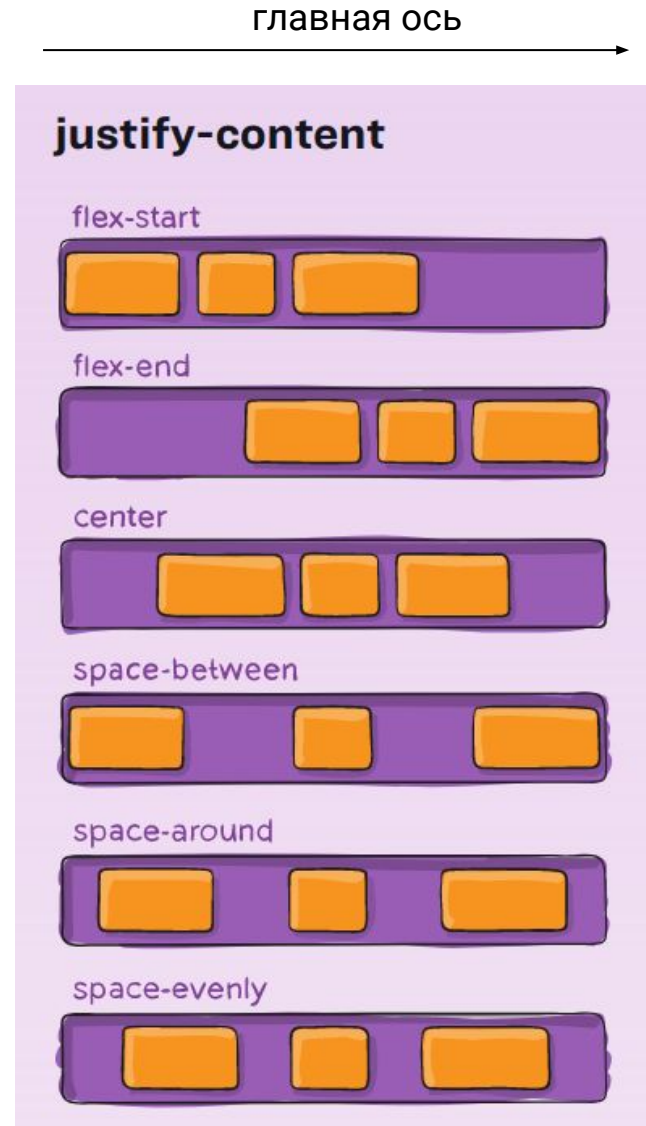


Flex

За это отвечает специальное свойство `justify-content`, выравнивающее элементы по основной оси (скриншот с веб-сайта css-tricks.com):

Попробуйте различные варианты и вы увидите, что нам подходит `space-between`:

```
.nav {  
  display: flex;  
  justify-content: space-between;  
}
```



изображение с css-tricks.com



Flex

Важно: вы должны понимать, что это не какие-то "космические" знания, которые берутся непонятно откуда, а просто кто-то придумал правила и инструменты (тот же flexbox) и мы учимся ими пользоваться так, как придумал автор (примерно так же мы учимся пользоваться любыми другими инструментами).



FlexBox Froggy

Мы предлагаем вам поиграть в специальную игру, которая поможет вам детально разобраться со свойствами, используемыми во Flexbox.

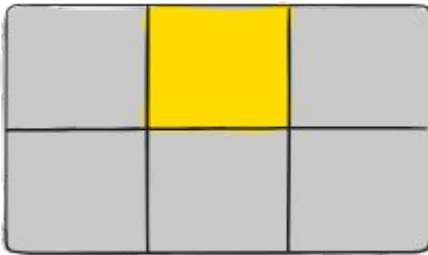
Адрес игры: <https://flexboxfroggy.com/#ru>



CSS Grid

CSS Grid – более мощная система, позволяющая более гибко (по сравнению с Flexbox) управлять размещением элементов.

В отличие от Flexbox, в котором вы управляете в основном, положением элементов в одной строке, Grid позволяет выстраивать сетку из строк и столбцов:



изображение с css-tricks.com



container, item, line, cell

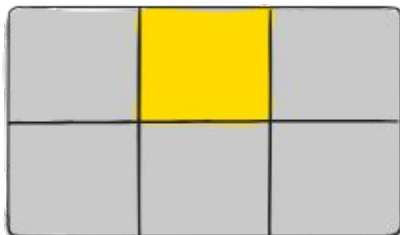
В рамках этой технологии используется уже 6 ключевых терминов:

1. Container (контейнер) – элемент, внутри которого мы хотим управлять размещением дочерних элементов
2. Items (элементы) – элементы, которые находятся внутри контейнера и размещением которых мы хотим управлять
3. Line – линии (горизонтальные и вертикальные)



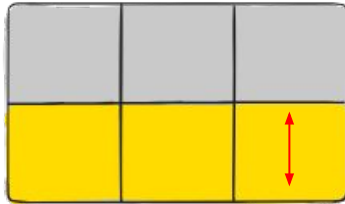
3-я линия по вертикали

4. Cell – ячейка

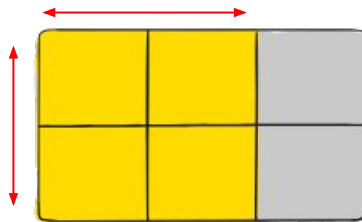


area & track

5. Track – строка или столбец между двумя Line:



6. Area – область, ограниченная 4-мя Line:



CSS Grid

Как вы видите, понятий гораздо больше, и возможностей, которые будут нам предоставлены также будет гораздо больше.



CSS Grid

Мы предлагаем вам поиграть в специальную игру, которая поможет вам детально разобраться со свойствами, используемыми во CSS Grid.

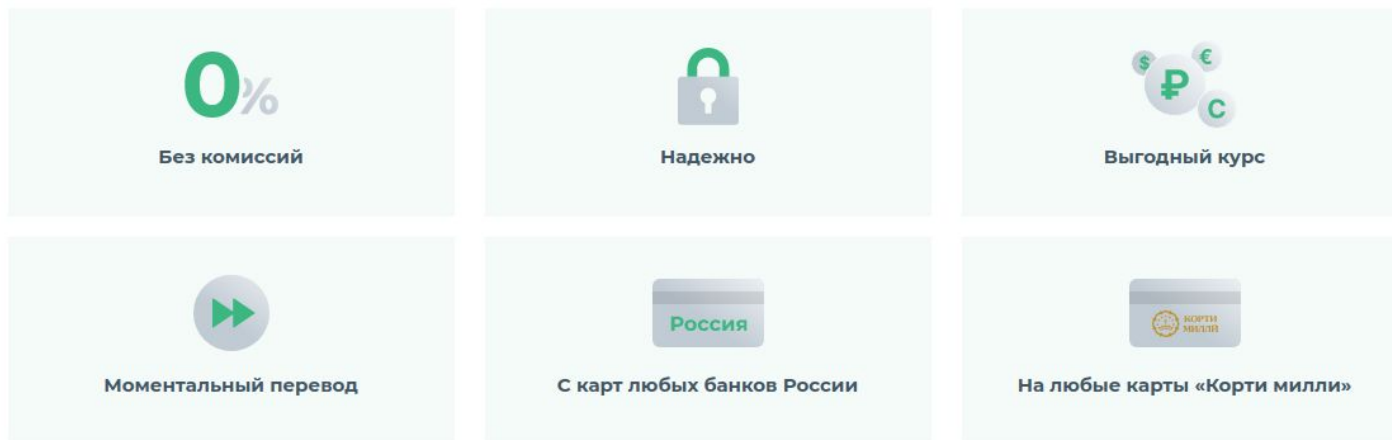
Адрес игры: <https://cssgridgarden.com/#ru>



Преимущества

Попробуйте самостоятельно "отверстать" блок преимуществ сначала с помощью Flexbox, а затем с помощью Grid. Можете делиться этим кодом в группе и обсуждать получившиеся решения.

Преимущества alif mobi



Преимущества

Вернёмся к преимуществам и обсудим вопрос картинок в блоках:

Преимущества alif mobi

Без комиссий



Надёжно



Выгодный курс



Моментальный перевод



С карт любых банков России



На любые карты «Корти милли»



Преимущества

Картинки здесь тоже оформительские, поэтому должны делаться через css (т.е. у каждого блока будет свой класс с заданной картинкой, например:

`.advantage-zero-commission`).

Но как тогда выставить текст? Очень просто, можете выставить фиксированный `padding/margin` сверху или спозиционировать сам текст относительно своего родителя.



Инструкция

Вот этот блок поинтереснее:

- 1 Скачайте кошелёк alif mobi и зарегистрируйтесь
- 2 Привяжите карту российского банка с которой будет осуществляться перевод
- 3 Введите номер карты «Корти милли» получателя и нажмите на кнопку «Отправить»
- 4 Получатель может обналичить деньги в банкоматах

Здесь всё хитро: за тип списка отвечает свойство `list-style-type`. Но там нет чисел без точек. Можно, конечно, вручную прописать их, но это неудобно.



Инструкция

Оказывается, можно отменить поведение по умолчанию, и назначить свою генерацию с помощью свойств `counter-reset` и `counter-increment`:

```
.instruction .steps {  
  counter-reset: number;  
  list-style-type: none;  
}  
  
.instruction .steps .step::before {  
  counter-increment: number;  
  content: counter(number);  
}
```

А затем уже стилизовать псевдоэлемент `::before` нужным вам размером шрифта и цветом.



Инструкция

Из оставшегося у нас:

- небольшая анимация на кнопках (при наведении)
- и анимация свёртывания/развёртывания для блока вопросов:

Если у получателя нет карты «Корти милли»



Если у отправителя нет карты российского банка



Формы, их стилизацию и валидацию (проверку на корректность с отображением ошибок) мы будем проходить уже на JS Level I.



Анимация



Анимации

В качестве бонуса (перед переходом к JS), давайте обсудим вопросы анимации. Многие анимации можно сделать только с помощью CSS (т.е. без использования дополнительных инструментов вроде JS). Но в большинстве случаев CSS будет использоваться совместно с JS. Поэтому в этой лекции мы обсудим ту часть, которую можно использовать без JS, чтобы уже на Level I, совмещать обе техники.



transition

Самый простой способ задать анимацию элементу – с помощью свойства **transition** (фактически, переход элемента из одного состояния в другое с анимацией определённого свойства или свойств).

Например, мы хотим, чтобы при наведении кнопка плавно уменьшала свою непрозрачность (**opacity**, которое по умолчанию равно **1.0**):

```
.primary-btn {  
  display: inline-block;  
  padding: 15px 30px;  
  border: 0;  
  border-radius: 4px;;  
  font-size: 18px;  
  font-family: inherit;  
  text-decoration: none;  
  color: var(--primary-fg-color);  
  background: var(--primary-bg-color);  
  transition: opacity 100ms linear;  
}  
  
.primary-btn:hover {  
  opacity: 0.8;  
}
```



transition

transition раскладывается на несколько составляющих:

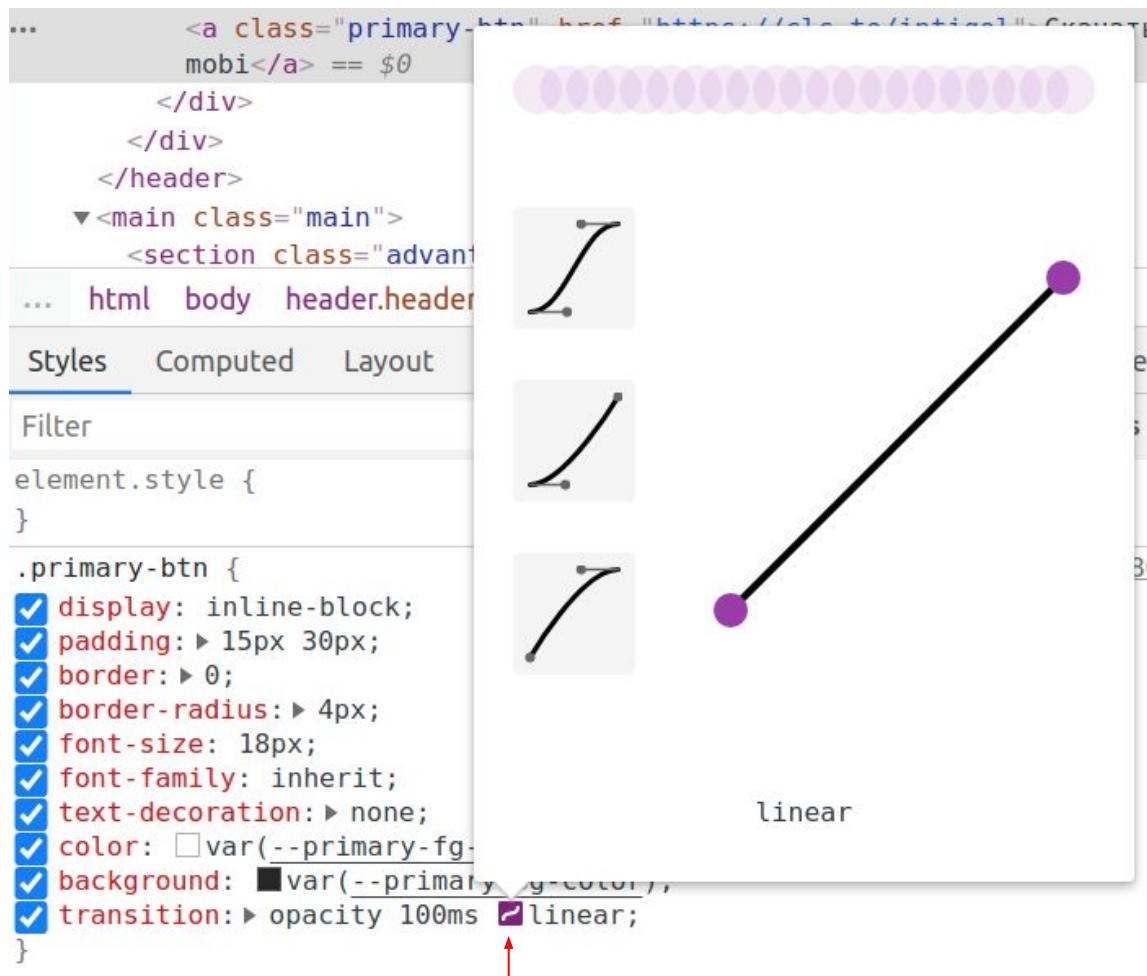
- **transition-property** – свойство, которое будем анимировать (чтобы анимировать все, нужно использовать значение **all***)
- **transition-duration** – длительность перехода, может указываться в секундах (суффикс **s**) или миллисекундах (суффикс **ms**)
- **transition-timing-function** – функция, по которой вычисляется форма изменения ([см. демонстрацию](#))
- **transition-delay** – задержка перехода (по умолчанию – **0**)

Примечание*: некоторые свойства не анимируются, например, **display** (поэтому **all** их затрагивать не будет).



transition

В панелике **Styles** Chrome вы также можете сразу настроить функцию перехода и посмотреть в действии:



animation

Для простых эффектов этого может быть достаточно. Но для более тонкого контроля используется – **animation**, который буквально позволяет вам указать ключевые точки (называется **keyframes**) состояния элемента, через которые будет проходить анимация.



animation

Демонстрация всегда лучше слов, поэтому предлагаем вам ознакомиться с целой библиотекой, в которой уже собраны разные эффекты: <https://animate.style>

Их реализации (этих эффектов), вы сможете найти на GitHub:

<https://github.com/animate-css/animate.css/tree/main/source>



Итоги



Итоги

В этой лекции мы рассмотрели приближенную к реальной практику сбора веб-сайта.

Конечно же, для профессиональной вёрстки, стоит достаточно много практиковаться и пробовать разные решения, но полученных нами знаний на этом курсе достаточно, чтобы переходить к JS (что и было целью курса).



ДОМАШНЕЕ ЗАДАНИЕ



Орг.моменты

Практикум состоит из 8 обязательных занятий. Начиная с 23 декабря мы выкладываем новые занятия каждый Пн в 10:00 (по Душанбе).

Каждое воскресенье в 23:59 (по Душанбе) дедлайн сдачи домашнего задания.

Если не успеете сдать в срок домашнее задания, тогда этот практикум будет для вас закончен и вы сможете зарегистрироваться на запуск следующего через несколько месяцев.

Все вопросы вы сможете задавать в [Телеграм канале](#).



ДЗ 1: flex

У бота будет документ:

```
<ul class="audience-list">  
  <li class="audience-item">Тем, кто хочет стать FullStack-разработчиком</li>  
  <li class="audience-item">Готовым не спать, стараться и выполнять все задания</li>  
  <li class="audience-item">Для людей без страха познавать много нового и решать проблемы</li>  
</ul>
```



ДЗ 1: flex

Используя flexbox, сделайте так, чтобы элементы были равномерно распределены:



Тем, кто хочет стать
FullStack-разработчиком



Готовым не спать, стараться
и выполнять все задания



Для людей без страха познавать
много нового и решать проблемы

Важно: нужны только правила для распределения элементов (цвет, текст, границы и всё остальное – не нужно).

Используйте минимум правил и минимум свойств в правилах.



ДЗ 2: simple grid

У бота будет документ:









```
<ul class="social-list">  
  <li>Facebook</li>  
  <li>Instagram</li>  
  <li>Телеграм</li>  
  <li>YouTube</li>  
  <li>TikTok</li>  
  <li>Одноклассники</li>  
  <li>LinkedIn</li>  
  <li>Вконтакте</li>  
</ul>
```

Важно: нужны только правила для распределения элементов (цвет, текст, границы и всё остальное – не нужно).



ДЗ 2: simple grid

Организируйте следующую раскладку:

 Facebook @alifbank.tj	 Instagram @alifbank.tj	 Телеграм @alifbank	 YouTube youtube.alif.tj
 TikTok @alifbank.tj	 Одноклассники @alifbank	 LinkedIn @alif-capital	 Вконтакте @alifbank



ДЗ 2: simple grid

Используйте:

1. `grid-template-columns` и `grid-template-rows`
2. Разберитесь со значениями `auto` и `fr`

Используйте минимум правил и минимум свойств в правилах.



ДЗ 3: complex grid

У бота будет документ:

```
<ul class="product-list">  
  <li class="product-item salom">Карта «Салом»</li>  
  <li class="product-item milli">Платёжная карта «Корти милли»</li>  
  <li class="product-item shop">Интернет-магазин alif shop</li>  
</ul>
```

Используйте только селекторы по классу (для родителя) и комбинированные (классы для родителя + ребёнок):

`.product-list {}`

`.product-list .salom {}`

и т.д.

Важно: нужны только правила для распределения элементов (цвет, текст, границы и всё остальное - не нужно).



ДЗ 3: complex grid


Используя CSS Grid реализуйте следующую раскладку:

2x

Карта «Салом»


Ею можно купить товары и услуги в рассрочку быстро, без документов и предоплаты

[Узнать больше >](#)



1x


Платёжная карта «Корти милли»



Интернет-магазин alif shop

Широкий ассортимент бытовой техники, мобильные телефоны и многое другое в рассрочку до 12 месяцев

[Перейти в магазин](#)





ДЗ 2: complex grid

Используйте:

1. `grid-template-columns` и `grid-template-rows`
2. `grid-column-start` и `grid-column-end` (используйте для них числовые значения)
3. Разберитесь со значениями `auto` и `fr`

Используйте минимум правил и минимум свойств в правилах.



Спасибо за внимание

alif skills

2022г.

